Applying the Dependability Paradigm to Computer Security

Catherine Meadows
Code 5543
Center for High Assurance Computer Systems
Naval Research Laboratory
Washington, DC 20375

Abstract

Dependability is that property of a computer system such that reliance can justifiably be place on the service it delivers [Lap94]. In this paper we contrast the way different ways faults are handled in the dependability paradigm with the way they are handled in the current paradigms for secure system design. We show how the current security paradigm is generally restricted to a subset of the types of approaches used in dependability, largely concentrating on fault prevention and removal while neglecting fault tolerance and forecast, and argue that this paradigm is fast becoming obsolete. We discuss the implications of extending the security paradigm to cover the full range of options covered by dependability. In particular, we develop a rough outline of a fault model for security and show how it could be applied to better our understanding of the place of both fault tolerance and fault forecast in computer security.

1 Introduction

Researchers in dependability have long followed a five-part approach to designing dependable systems. Briefly, system failure is prevented by identifying the kinds of faults that can lead to failures and then seeking either to prevent faults, tolerate faults, identify and remove faults, forecast faults, or apply some combination of the above. In [Lap94] these are defined as follows. Fault prevention is the prevention of occurence or introduction of faults in the first place, fault tolerance is the ability of a system to operate without failure in the presence of faults in some of its components, fault identification and removal is the identification and removal of faults in systems, and fault forecasting is the estimation of the incidence of present and future faults, as well as their consequences.

Security researchers, on the other hand, have tended to concentrate on only half of this approach. A considerable amount of work has been done on fault prevention and fault identification and removal in secure systems, as well as a certain amount on fault classification, but, with few exceptions, such as [LBF$^+$93], relatively little on fault tolerance or forecast exists. This is partly because these problems are much harder in computer security, which must deal with preventing failures caused by intruders whose motivation and capacities may be difficult to estimate. It is hard to predict the consequence of a fault, or to determine in what cases a system can be tolerant of it, when the environment itself is unpredictable. Thus, when security is modeled, it is usually assumed that the system is trying to operate in the face of a hostile adversary with unlimited capabilities in certain areas. For example, the modeling of secure operating systems assumes the existence of Trojan Horse code that is able to signal information along covert channels, while the modeling of secure protocols assumes the existence of an intruder who is able to read and modify all traffic, gain control of nodes, and compromise old secret information. In the actual development of secure systems, such assumptions may be relaxed, of course, but, for theoretical models the most stringent assumptions usually apply.

In [Mea94] we argued that this paradigm was fast becoming obsolete, that it was time to start moving away from worst-case assumptions, and that we should try approaching this problem by developing a fault model that allowed the modeller of a secure system to take different types of assumptions into account. Here we reproduce and expand upon the reasons we gave in that paper: first, that the theories devoted to the more traditional computer security problems are reaching the stage of maturity where such models are necessary, secondly, that is impossible even to consider some of the security problems that are emerging without at least the beginning of such a model, and thirdly, that the growing complexity and interconnection of systems would result in overly restrictive constraints on connectivity and services if the worst-case assumptions were always applied.

We begin with the issue of maturity. The concentration on worst-case assumptions is characteristic of a developing theory; one's first goal is to test the limits of the theory by applying it to the most severe cases possible. It is also usually paradoxically the case that it is easier to develop theories under worst-case assumptions; worst-case assumptions are usually the simplest to develop or formulate. However, as a field matures, the limitations of this approach become more clear. Theories developed for worst-case assumptions may turn out to be impractical to apply. Moreover, it may also turn out that, as we delve more deeply into the problem, that it is always possible to extend our notion of a "worst-case" assumption so that our model fails to handle it. Such has been the case in natural language processing, for example; it is always possible

Report Documentation Page		*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **1995**	2. REPORT TYPE	3. DATES COVERED **00-00-1995 to 00-00-1995**
4. TITLE AND SUBTITLE **Applying the Dependability Paradigm to Computer Security**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Research Laboratory,Center for High Assurance Computer Systems,4555 Overlook Avenue, SW,Washington,DC,20375**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**5**	

to come up with some kind of bizarre formulation that a given theory cannot handle but that is nevertheless natural language.

We are beginning to reach this point in the theory of secure systems. This has been particularly noticeable in the recent developments of theories of information flow. For any theory that has been developed, it has been possible to come up with a type of covert channel that can defeat it. This has culminated in theories that are able to handle such subtle notions as probabilistic channels that convey information by varying the probability that events will occur [Gra91].

Such theories are difficult to apply, both because of the difficulty of calculating the probabilities of events in a realistic system, and because of the amount of variables that must be considered. Perhaps at this point it is time to re-evaluate these information flow theories by developing realistic fault models for covert channels and evaluating theories in terms of these models.

The second reason is the changing emphasis of security research. In the past research in secure computer systems has concentrated on secrecy. In theory at least, as long as one has sound non-bypassable access controls and has eliminated all covert channels, there is no way that secrecy can be compromised in a system, even if all code not entrusted with enforcing the security policy is actively attempting to subvert it. This is not the case with other security attributes such as integrity or denial of service. An access control mechanism can decide which processes modify what data items, and in what order, but whether the data items are modified correctly is the responsibility of the processes themselves. Likewise, an access control mechanism can assist a process in providing a service by preventing other processes from denying it accesses to resources such as memory, but it cannot guarantee that that process will actually provide the service. Thus, if we assume that processes are always actively attempting to subvert the security policy this will not result in a meaningful model of a secure system. On the other hand, we do not have the resources to guarantee that all processes are completely trustworthy. We will need some notion of different degrees of trust. A fault model can assist us in developing such a notion by giving us a means of characterizing the different ways that a process can attempt to subvert the security policy, as well, possibly, of a means of ranking these different kinds of attacks in order of their likelyhood and difficulty.

The third reason for the need for a new approach is the growing complexity and interconnection of computer systems. In many cases a system may be connected to another system and rely upon its services even though the second system is not completely trustworthy. Assuming all such systems are completely untrustworthy will result in grossly overrestrictive limitations on connectivity and services. Thus we need some means of identifying the ways in which the different components of a large distributed system can fail. We can be helped in this by the fact that there already is a large body of work in fault-tolerance that addresses cases in which the a portion of the components of a system behave in an arbitrarily malicious fashion, starting with [LSP82]. However, this model needs to be refined further to take into account the specific different ways in which a malicious node can misbehave, as well as the specific countermeasures.

2 Outline of a Fault Model for Security

In general, we can divide the types of security faults a system may be subject to into three areas. The first includes faults in the security mechanisms themselves. These would include covert channels in a multilevel system, a cryptosystem that can be broken under different circumstances, or the many well-known Internet security holes, as described for example by Cheswick and Bellovin [CB94]. In many cases, for example covert channels, complete elimination of these faults may not be possible if the other goals of the system (in this case performance) could be met. In other cases, use of the faulty system may be so entrenched that, although in theory it could be replaced, in actuality it would not be possible to do that without a great deal of expense. This is particularly true for security, since in many cases the amount of damage that can be caused by a security vulnerability is not well understood until the system has been in use for some time and is well established. One well-known example is the vulnerability of personal computers to viruses. Personal computers were originally implemented without access controls because they were intended to be single-user machines. Likewise, users of the personal computers did not understand the risks of using potentially hostile software. The designers and users of the personal computer did not adequately realize that software, too, can be considered a "user" whose access should be limited.

The second area includes hostile attacks on the system itself. These include but are not limited to break-ins to system components, compromise of secret data such as encryption keys, eavesdropping on message traffic, impersonation of legitimate principals, modification of data, Trojan Horse code in multilevel systems, and so forth.

The third area is somewhat less obvious than the other two. This area consists of the various different non-malicious, but nevertheless risk-producing ways in which humans can use a secure system. These include things such as bad choice of passwords, incorrect setting of security parameters, downloading object code from public bulletin boards, entrenchment of systems with known security problems, and so forth. Although this class of fault has not been given as much attention as the first two, a recent study of British banking practices [And93] has shown that most security failures resulted from just this type of incorrect use. Schneier [Sch94] has also shown how the many of the vulnerabilities in the cryptographic mechanisms in use today result as much from human error and misuse as from any problems in the mechanisms themselves.

A security failure will generally be the result of the interaction of a number of faults from the different classes. Consider viruses in personal computers, for example. Viruses are written by malicious humans, and so may be considered an example of the second kind of fault, hostile attack. However, viruses have only been as successful as they have been because of the weak security controls in personal computers. This is an example of the first kind of fault: inadequate security mechanisms. Finally, the virus problem is exacerbated by careless use of software whose origin is unknown or suspicious. Thus the security failure of virus infection can result from the interaction of three different faults.

We note that this taxonomy corresponds roughly to the taxonomy of human-made faults compiled by IFIP WG 10.4 [Lap94] in their work on developing a terminology for dependable systems. These are divided into design faults, which correspond to our notion of faults in security mechanisms, interaction faults, which corresponds to our category of human misuse, and finally malicious logics and intrusions, which together correspond to our class of hostile attacks, although Laprie et al. go further and divide the classes into internal attacks (malicious logic) and external attacks (intrusions). Thus it seems likely that our taxonomy could be straightforwardly integrated into the more general fault model used by the dependability researchers.

We note that out taxonomy at this point is relatively rough and lacking in detail. It should not be too difficult to flesh it out further, however. Indeed, a fair amount of work already exists on this kind of topic. For example, Landwehr et al. [LBMC94] present a taxonomy of computer security flaws from the point of view of fault detection and prevention that might also be useful for our purpose. In [CB94] Cheswick and Bellovin give a classification of the different kinds of attacks on the Internet. The COAST project has also been developing taxonomies of such areas as intrusions and Unix security faults, again from the point of view of fault (in this case, intrusion) detection [Asl95, Kum95]. These works and others could give us a good starting point for our taxonomy.

3 Fault Forecast and Security

Assessing the security of a system in any meaningful way can be a difficult problem. Most criteria for the evaluation of secure systems, for example the TCSEC [DoD85], concentrate on specifying the techniques that are used to develop the system, not on the degree to which the systems actually previde protection against security failures. But, as Littlewood et al. [LBF+93] point out, although implementing such requirements may be helpful in facilitating the development of secure systems, they do not provide any help in giving an operational measure of computer security in the sense that there are operational measures of reliability in terms of such measures as mean time between failures or time until first failure. Yet such measures would give a much better idea of the degree of security a system would provide. The problem, of course, is that such measures are very difficult to devise for secure systems.

Our position is that, although there are aspects of the security problem that are very difficult to measure, a well-thought-out fault model can help by classifying faults in a way so that they are sorted out according to how they can be measured. We can then concentrate on measuring properties of the more quantifiable faults, while making estimates for the less quantifiable. Here we show how the rough taxonomy we have developed would aid us in this.

We first consider faults in the design of the security mechanisms themselves. Although in some cases we may worry about the likelyhood that a security fault will come to light, in many cases the fault is already well understood. In these cases we can measure the fault in terms the amount of access it can give an opponent, and in terms of the amount of effort it takes to take advantage of the fault. Many of these measures are straightforward, such as the capacity of a covert channel, the amount of time it takes to break a cryptoalgorithm, or the degree of unintended access that can be given by a poorly designed access control system.

Next we consider non-malicious human-induced faults. Although human behavior is harder to measure than the difficulty of cracking a particular security barrier, it is still possible in many cases to estimate the likelyhood of a non-malicious fault occurring. This is because many of these faults are the result of users choosing the most convenient way of doing things, which can generally be considered fairly predictable. Indeed, a number of studies have been performed of such behaviors as the ways in which passwords are chosen and the ways in which viruses are spread by users of personal computers [KW93]. If we consider faulty implementation a non-malicious human-induced fault, then we can also use the considerable body of work that has been devoted to producing measures of software reliability.

Finally, we consider malicious attacks. These are the hardest to quantify. Generally, we would measure an attacker in terms of resources available, including intelligence, computing power, and time, the willingness to expend these resources, which would be measured in terms of such things as the payoff of mounting the attack or whether or not there are more vulnerable system available, and finally, what the goals of the attacker are. The first tells us whether or not an attacker is capable of mounting an attack and how, the second tells us whether or not the attacker is likely to want to mount the attack, and the third tells us the amount of harm the attacker is likely to cause if the attack succeeds. For example, Littlewood et al. [LBF+93] discuss modeling operational security in terms of in terms of reward (to attackers) as a function of effort expended, which would correspond in our taxonomy to a measure in terms of the first two fields.

The trouble is, of course, is that many of these quantities are very difficult to measure, and, indeed, in cases in which figures do exist, neither attackers nor victims are very willing to share them. Thus, reliable figures on this are very hard to find. The best we can do in most cases is to say that a particular system provides a certain degree of security against an attacker which a certain amount of available resources and a certain degree of motivation. However, this in itself can be useful to someone who is determining whether or not to rely upon a particular system for a particular application.

4 Fault Tolerance and Security

Security has made more use of fault tolerance than is perhaps realized. Multilevel secure operating systems are designed to be tolerant of Trojan Horse code in the untrusted parts of the system. Key distribution protocols are designed to be tolerant of dishonest principals and compromise of session keys. Secure database management systems are designed to be tolerant of those who are trying to infer sensitive data. Secret sharing schemes are designed to be secure against dishonest trustees who may betray their shares of the secret. However, the types of faults tolerated by secure systems have tended to fall for the most part into the class of malicious attacks, while ignoring human misuse and flaws in security mechanisms. Moreover, the malicious attacks considered tend to fall into the category of worst-case scenarios.

This is slowly begining to change, however, particularly in the class of human misuse. To give a relatively small example, a growing body of work is beginning to appear on authentication protocols that are tolerant of poor password choice [LGSN89, BM92]. Likewise, in [Sch94] Schneier describes a number of design heuristics for cryptographic algorithms which would make them easier to implement and use by mistake-prone programmers, system managers, and users. In [And93] Anderson also points out the necessity of designing simple equipment that can be operated by an inexperienced user, rather than more flexible but complex equipment that can only be operated by experts. Some attention is also beginning to be paid to tolerance of faulty security mechanisms as well; for example in [Sch94] Schneier recommends the use of design diversity and use of multiple encryption algorithms to counteract possible flaws in the algorithms or their implementation.

We also note that it is possible to develop non-worst-case fault-tolerant mechanisms for malicious attacks as well, if one takes the psychology of the attacker into account. An example would be the use of "honeypots" to distract intruders away from genuinely sensitive data while providing system administrators time to monitor them, as was done in [Sto89][1].

Although attention is starting to turn to tolerance of misuse and faulty mechanisms as well as active attacks, the work on this subject is still scattered throughout the literature and not as yet organized into a coherent whole. We have shown how even our rough fault model has been able to provide some insight into the application of the fault-tolerance to computer security; a more detailed and fleshed-out model should be able to provide much more.

5 Conclusion

In this paper we argued that the paradigm of concentrating fault detection and prevention in computer security was fast becoming inadequate, and that it was time to pay more attention to fault tolerance and prediction as well. We have outlined some of the challenges faced in doing so, and have constructed a rough fault model that assists us in thinking about these challenges. A more detailed and fleshed-out model should be of even more use in helping us grapple with these problems.

[1] I am grateful to one of the anonymous referees for pointing out this strategy as an example of fault tolerance.

References

[And93] Ross Anderson. Why crypto systems fail. In *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 215–227. ACM SIGSAC, ACM, Nov. 3-5 1993.

[Asl95] Taimur Aslam. A taxonomy of security faults in the unix operating system. Master's thesis, Purdue University, August 1995.

[BM92] S. M. Bellovin and M. Merritt. Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks. In *Proceedings of the 1992 IEEE Computer Society Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, May 4-6 1992.

[CB94] William Cheswick and Steven Bellovin. *Firewalls and Internet Security: Repelling the Wily Hacker.* Addison-Wesley, 1994.

[DoD85] Department of Defense Trusted Computer System Evaluation Criteria. DOD 5200.28-STD, August 1985.

[Gra91] J. W. Gray III. Toward a Mathematical Foundation for Computer Security. In *Proceedings of the 1991 IEEE Symposium on Security and Privacy*, pages 21–34. IEEE Computer Society Press, May 1991.

[Kum95] Sandeep Kumar. *Classification and Detection of Computer Intrusions.* PhD thesis, Purdue University, August 1995.

[KW93] Jeffrey Kephart and Steve White. Measuring and modeling computer virus prevalence. In *Proceedings of 1993 IEEE Symposium on Security and Privacy*, pages 2–15. IEEE Computer Society Press, May 1993.

[Lap94] Jean-Claude Laprie, editor. *Dependability: Basic Concepts and Terminology.* IFIP WG 10.4, August 1994. draft.

[LBF$^+$93] Bev Littlewood, Sarah Brocklehurst, Norman Fenton, Peter Mellor, Stella Page, David Wright, John Dobson, John McDermid, and Diter Gollmann. Towards operational measures for computer security. *Journal of Computer Security*, 2:211–229, 1993.

[LBMC94] C. E. Landwehr, A. R. Bull, J. P. McDermott, and W. S. Choi. A taxonomy of computer security flaws, with examples.

ACM Computing Surveys, 26(3), September 1994.

[LGSN89] T. M. A. Lomas, L. Gong, J. H. Saltzer, and R. H. Needham. Reducing Risks From Poorly Chosen Keys. *Operating Systems Reviews*, 23(5):14–18, 1989.

[LSP82] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.

[Mea94] C. Meadows. The need for a failure model for security. In *Proceedings of Dependable Computing for Critical Applications 4*, 1994. to appear.

[Sch94] Bruce Schneier. Designing encryption algorithms for real people. In *Proceedings of the 1994 New Security Paradigms Workshop*, pages 98–101. ACM SIGSAC, IEEE Computer Society Press, August 3-5 1994.

[Sto89] Clifford Stoll. *The Cuckoo's Egg*. Doubleday, 1989.